

MorteSense

DIY Home Security

Shohin Abdulkhamidov

Dept. of Computer Science

San Jose State University

San Jose, United States

shohin.abdulkhamidov@sjsu.edu

Diego Cruz

Dept. of Computer Science

San Jose State University

San Jose, United States

diego.cruz@sjsu.edu

Diego Garcia-Carrasco

Dept. of Computer Science

San Jose State University

San Jose, United States

diego.garciacarrasco@sjsu.edu

Spartak Gevorgyan

Dept. of Computer Science

San Jose State University

San Jose, United States

spartak.gevorgyan@sjsu.edu

Faramarz Mortezaie

Dept. of Engineering

San Jose State University

San Jose, United States

faramarz.mortezaie@sjsu.edu

Abstract—In recent years, DIY security system technologies have been simplified and became more affordable for the general public. This coincided with the emergence of brands like Ring, Blink, Wyze, SimpliSafe, and Vivint, which specialized in smart home security. These brands offer a range of products, including front-door cameras, infrared motion sensors/detectors, and in-home cameras providing various views of the house.

One notable drawback of these product ecosystems is their reliance on a mobile application and a smartphone capable of running it. If the application ceased to receive support or the company went out of business, the associated hardware might no longer function as intended, resulting in unnecessary e-waste and unsustainable security practices.

The project addressed the dependency on a smartphone application by utilizing SMS text messaging within a modular motion detection security system. The system is part of an ecosystem that could be customized to meet the user's specific requirements and specifications. It was designed with sustainability in mind, capable of persisting even if manufacturer support became unavailable.

Index Terms—DIY, Home Security, Microcontroller, Open-Source, L^AT_EX

I. PROJECT OVERVIEW

A. Project Goals and Objectives

The MorteSense DIY security system, more accurately described as a Modular Microwave Motion Security System with mobile notifications, was a project intended to provide efficient and reliable security for homes and businesses. The system uses microwave motion sensors to detect movement within a certain range and sends a mobile notification to the owner's phone in case of any suspicious activity.

There was an emphasis placed on the open documentation with the project group working closely with the advisors to allow for prospective end users to take the system design and modify it to their own needs and specifications, along with

allowing some degree of freedom to enable end users to add their own modules such as cameras, passive infrared (PIR) sensors, and piezoelectric sensors.

B. Problem and Motivation

The MorteSense project aims to address the growing need for accessible, cost-effective, and reliable security solutions for residential and commercial properties. Traditional security systems often come with proprietary modules, firmware, software, or communication protocols, making them impractical for end users that desire system flexibility.

The motivation behind MorteSense was to create a DIY security system that leverages microwave motion detection technology to provide an affordable, easy-to-install, and efficient building block towards monitoring and securing properties. This project is important as it can pave the way to make more advanced security systems accessible to a wider audience, thus improving safety and security for individuals and businesses.

To ensure real-time alerts, the system incorporates mobile notifications, which enable swift responses to potential security breaches by allowing for the use of different means of communication. This approach effectively addresses the need for a more convenient and user-friendly security solution.

C. Project Application and Impact

In the Age of Information, advanced security systems have become crucial in ensuring public health, safety, and welfare. With growing concerns regarding security, the need for efficient and reliable security systems is more pressing than ever. This system has been designed with critical factors such as global, cultural, societal, environmental, and

economic considerations in mind, ensuring accessibility and effectiveness for a wider range of users.

Public health, safety, and welfare are essential considerations for any security system. The microwave motion security system with mobile notifications can detect and notify homeowners of potential security risks through different means, which is vital for public safety. The mobile notification feature allows prompt action to be taken, reducing the risk of harm and property damage. By reducing crime rates in a neighborhood, the security system can also positively impact the safety and well-being of the community. In case of a security breach, the system can notify emergency services which minimizes damage and protects lives.

II. BACKGROUND AND RELATED WORK

A. Background and Technologies

In recent years, DIY home security has risen in popularity, with modern solutions including doorbell cameras, fingerprint-scanning door modules, infrared motion detection sensors, and the occasional surveillance camera. Companies like Ring, Blink, Wyze, SimpliSafe, and Vivint have had a significant impact on improving the availability of home security and in the development of smart homes that are connected to the Internet of Things (IoT).

Leading up to this point, contemporary literature has explored how using different parts from different manufacturers in unique configurations can be integrated into IoT [1]. Additionally, current literature explores how DIY home security will develop in relation to the blockchain [2] and the growing influence of machine learning [3]. In light of this related work, this project entailed creating a DIY security system that will utilize contemporary microcontrollers and microwave motion detection sensors and will be able to interface with mini-motion proximity cameras. Modular design principles were used to create an efficient, small-scale security system that can add different access points through different modules. Combined with extensive documentation, users are able to test, maintain, and repair the system as needed.

In addition to using modular design principles, modern technologies including using Wi-Fi to connect to remote servers and notify users of events with SMS text messaging as well as web protocols to broadcast the history of detections/captures and enact security network management, allow for an open ecosystem that can be tailored to the user's requirements/specifications and remain sustainable by persisting even if direct manufacturer support is no longer available.

In the context of this project, the DIY security system builds upon the work established by research conducted by IEEE engineers and scientists that used the Arduino and Raspberry Pi platform. This project explored cutting-edge devices and technologies that were not necessarily present

at the time of previous studies, and further analysis is given in the State-of-the-Art section following this literature review.

Komninos et al. explores the threats to smart homes and smart grids with respect to software security and attacks involving invasions. This study uncovered that smart grids have the threat of message modifications, replay attacks, metric impersonation, and denial-of-service, while smart homes have similar threats, with unique ones including false synchronization and eavesdropping [4].

The main concerns for current smart homes now include false synchronization and denial-of-service, being the most convenient methods for burglars to effectively invade and loot a home. The two aforementioned attacks were taken into account when developing the DIY security system, learning from the limitations of this particular study.

A 2017 study conducted by two IEEE researchers explored the vulnerabilities that existed in DIY home security systems that used sensors, microcontrollers, Raspberry Pi, and ZigBee communications [5].

Jose and Malekian found that one limitation of these systems at the time included the lack of an algorithm used to understand typical user behavior and tripping an alarm when the algorithm judges the detected behavior as erratic. Another limitation of systems at the time was that the ZigBee and IEEE 802.15.4 communications protocol were susceptible to replay attacks, allowing attackers familiar with the technology to exploit the vulnerability and intrude into the home [5].

Arif et al. examined the development of blockchain technology and its ability to handle great demand in its use cases. They evaluated the possibility of using blockchain technology to back IoT devices in DIY smart home security with a proposed blockchain configuration (see Appendix D). They found that while it was cost-effective and remained independent of cloud storage, the computation difficulty was set to the lowest possible, which may become complicated as more devices are added to the security system [4]. Variable computation difficulty is a new technique gaining popularity, but further research was made on the hardware requirements, and it was found to require more resources than what had been planned for the scope of the project. This is because the project will include web services and a deliberately designed user experience.

IEEE member Qusay Sarhan conducted a comprehensive literature review in 2020, providing an overview of dozens of research reports on smart home safety and security systems that used the Arduino platform. It was found that the most significant challenges to building these systems included physical attacks, device failure (of microcontroller/module), power/internet outages, and software compatibility [1].

These are factors that weighed heavily on this project and provided a point of reference for further learning and

improving the work that has already been done in the field. Additionally, the review provided a few subjects of interest, including extendability, performance, visualization, and testing for stress/robustness [1].

Lastly, Khan et al. built on the work conducted by Arif et al. by exploring the role that machine learning could play in home security systems that utilize blockchain technology. They found that using a Deep Extreme Learning Machine (DELM) combined with blockchain could prove to be far more efficient compared to other algorithms [3]. Using machine learning was beyond the scope of this project, but it is a valid consideration in building upon the work for a graduate research project.

B. State of the Art

The current state-of-the-art home security systems employ different techniques and products to create a comprehensive system that can typically be tailored to the end user through the use of modules. These products are often well-established and are only revised in ways that are not directly observable by the end user. Rather, the underlying technology may change while the enclosure receives minor modifications at most [1]. The most common implementation for state-of-the-art home security systems involves the use of modules that connect wirelessly to a central unit through some wireless protocol [5].

Alternatively, each individual module can connect to the internet [1].

From there, the modules are handled by a mobile application that allows the user to configure notifications for detection, remotely set the alarm based on activity in the household, or remotely activate the alarm from a safe place outside the house if there is dangerous activity observed within [5].

Most modules are typically motion sensors or cameras, with motion sensors having had much more development over the past few years compared to cameras. For the past decade, motion sensors in DIY home security systems have used Passive Infrared (PIR) technology to detect motion with changes in temperature induced by moving objects compared to the ambient environment [1]. However, one limitation of PIR sensors is that they cannot detect motion past objects, which means that large furniture may prevent detection. Another limitation of PIR sensors is that their perception of temperature may be thrown off by high ambient temperature or by wind, meaning that if the weather is hot, it may not detect motion as a result of a false negative, or if it is windy, it may create a false positive [1]. In short, PIR sensors have a reduced effective distance because of the environmental factors that can affect their ability to make accurate detections. Nonetheless, the latest advancements have resulted in the recent acceptance of microwave motion detector sensors

that utilize the Doppler effect to detect motion behind objects and are not susceptible to the environmental factors that impact PIR sensors [1]. This means that cutting-edge DIY home security can more reliably detect motion behind large furniture and can be installed outside the house without as much concern for environmental factors affecting the capacity to detect motion.

The Raspberry Pi and Arduino platforms have been extensively used as research tools by the IEEE to explore different wireless communications technologies in the DIY space, as well as for a central DIY security platform that is compared to established home/personal security products [1]. Over the past decade, Arduinos and their respective microcontrollers have increasingly used different connectivity types such as XBee, Ethernet, and Wi-Fi, most recently [1]. With the release of the 802.11n-capable Raspberry Pi Pico W in 2022, the project builds upon the work covered by previous research that used both mainline platforms or similar microcontrollers.

III. PROJECT REQUIREMENTS

A. Domain and Business Requirements

The MorteSense project had several requirements that the development team met to create an efficient and reliable security system. These requirements included the system's ability to detect motion accurately and differentiate between human and non-human motion to avoid false alarms. Optional requirements included distinguishing between different types of human motion, web-based monitoring and control, and a built-in camera. The system's non-functional requirements included being reliable, secure, easy to use, and having a long lifespan with minimal maintenance.

B. System Functional Requirements

Below are the functional requirements deemed essential for the project:

- The system shall be able to detect motion accurately using microwave sensors with a range of at least 10 meters
- The system shall be able to send SMS notifications to a specific mobile device within 5 seconds of detecting motion
- The system shall have a built-in algorithm to differentiate between human motion and other types of motion, such as pets or moving objects, to avoid false alarms
- The system shall have a remote control that allows users to arm and disarm the system easily
- The system shall have a backup power supply that can provide at least 24 hours of continuous operation during power outages

The only functional requirement that was deemed desirable for the project involves the system having a web-based

interface that allows users to remotely monitor and control the system, view motion event history, and adjust system settings.

C. System Nonfunctional Requirements

Below are the nonfunctional requirements deemed essential for this project:

- The system shall be reliable and have a low false alarm rate of no more than 1
- The system shall be easy to install and set up with clear instructions and user manuals provided
- The system shall be secure and prevent unapproved entry to private data or system configurations

Below are the nonfunctional requirements deemed desirable for this project:

- The system should have a user-friendly interface that is intuitive and easy to use, with clear visual and auditory feedback provided
- The system should be design to have lower power consumption, consuming no more than 5 watts per hour to reduce operating costs

D. Context and Interface Requirements

With respect to interaction design, the team incorporated a user-friendly interface, clear instructions, and collected feedback from the advisor and cohorts to ensure that the security system was accessible to a broader audience, including older or less technologically-savvy individuals, making it more effective and better-suited to the intended audience.

E. Technology and Resource Requirements

There are several different resource requirements, and have been compiled into three different categories: Hardware, Software, and Miscellaneous.

With respect to hardware, the resources used include a Raspberry Pi Pico W Microcontroller with a 2.4GHz Wi-Fi Module furnishing, a 5.8GHz Microwave Motion Sensor, and the Thonny IDE with Micropython. The Microcontroller is programmed to connect to the home Wi-Fi network and send API requests to the backend server after processing motion detection data. The motion sensor emits electromagnetic radiation that penetrates through non-solid objects and bounces off solid objects. The Doppler effect is used to detect the slightest movement changes and the sensor makes use of serial communications to exchange data between the microcontroller and the motion sensor. This allows modification of sensitivity, distance, and other parameters of the sensor. Thonny allows for compiling of Micropython into C code to effectively program the hardware.

With respect to software, the resources used include the Python Flask framework for the database server, ReactJS for the frontend, the Twilio API for notification service, and

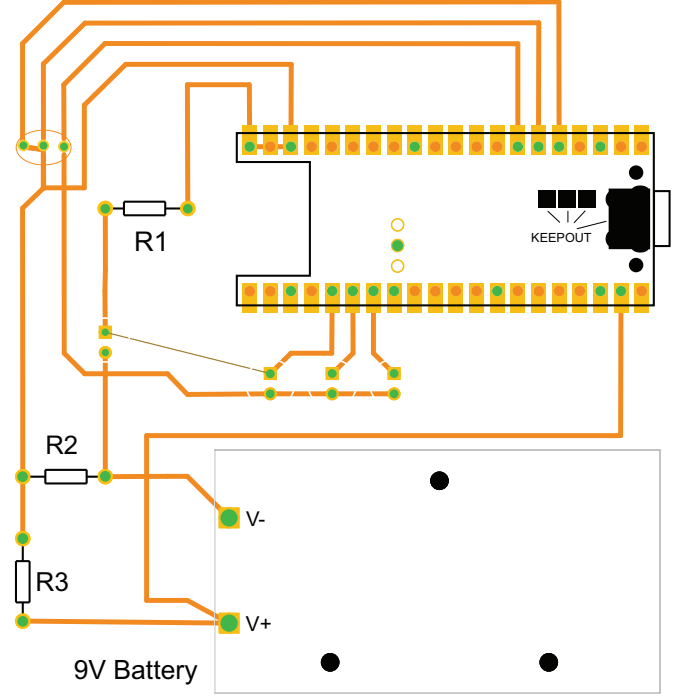


Fig. 1. PCB Diagram

Amazon AWS for cloud computing and reporting analysis. The Flask framework responds to the provided hardware API requests and provides a web-based interface to allow users to remotely monitor and control the system. The server functions as a gateway between AWS services and a SQLite database to store and provide systems information. The Twilio API is responsible for sending SMS notifications to client devices, and Amazon AWS is responsible for handling and analyzing data traffic to provide different metrics for what the sensor(s) report.

In terms of miscellaneous resource requirements, the Fritzing circuit design software, SolidWorks CAD software, soldering tools, and 3D printing hardware were used to design, simulate, and improve upon the hardware at different stages of assembly. Fritzing was used during ideation to design and simulate the hardware scheme before actual assembly and soldering. SolidWorks was used to design and prototype enclosures for the component assembly, and was aided by the Makerspaces available at San Jose State University.

IV. SYSTEM DESIGN

A. Architecture Design

1) *Project Hardware Architecture:* The PCB (Printed Circuit Board Diagram) diagram depicts the layout of the DIY security circuit board. It shows the placement of components and their interconnections on the board, as well as any

necessary information related to the fabrication and assembly of the board. While a circuit diagram provides a schematic representation of the electrical connections between components, the PCB diagram shows the physical layout of the components and the soldering points between them and the soldering layers. It is important for the efficient routing of traces and connections and will be used during the fabrication process.

The PCB will have 3 main components, the Raspberry Pi Pico W Microcontroller with 2.4Ghz Wi-Fi Module which is in the right top corner, the 9V Battery input in the right bottom corner, and finally, it will have the 5.8Ghz Microwave Motion Sensor which is in the left top corner of the board. The motion sensor emits electromagnetic waves which are then reflected back to the receiver and analyzed. If the waves are altered that means the object that reflected them is moving. The board also will have 4 LED indicators for showing the WI-FI connection indicator, power indicator, motion detection indicator, and connection to backend service indicator. Rx-Tx serial communication will be used between the microcontroller and motion sensor, for all other communication GP pins will be used.

The hardware has two main components, the motion detector, and the microcontroller. The microcontroller uses Rx-Tx serial communication to send and receive information into the motion detector in the form of bits, which can be used to send and receive instructions. Microwave Sensor out signal is used to detect the motion itself. As the microcontroller comes with an integrated Wi-Fi module, it will be used to communicate with the backend by sending and receiving API requests. The device will enter sleep mode and only be activated during motion detection, this will allow us to lower the power consumption significantly. All the hardware will be programmed in C and Micropython.

The microcontroller has an integrated server with web UI, which allows users to connect to it with their home devices such as smartphones, and do an easy setup of the systems, such as the Wi-Fi module. UI also provides basic information regarding the status of the motion detection device, battery lifetime, and other necessary information. The microcontroller has full control over the motion detector which allows users to do the precise configuration of system parameters, such as motion detection range, sensitivity, etc.

2) *Project Software Architecture*: The Notification Service is responsible for processing and sending SMS notifications to users when the motion is detected. It receives the motion detection information through requests from sensor management software and sends SMS notifications to the registered mobile numbers. Notification Service will integrate SMS gateway using Twilio API for communication with Client Devices. A database is required to store information about the user systems, such as user information, sensor

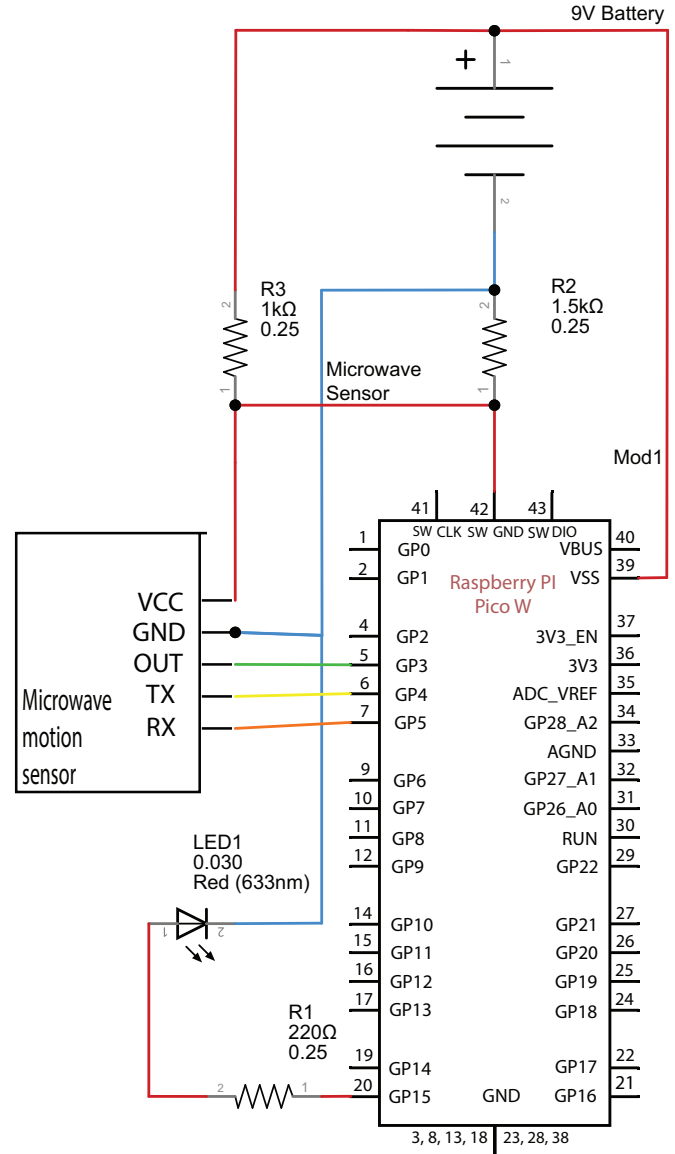


Fig. 2. Project Hardware Architecture

information, and notification settings. The Reporting and Analytics component is responsible for generating reports on the system's performance, such as the number of motion detections per day and average response time. It will help to analyze and further improve the quality of the systems.

User Management Service (UI) will provide an interface for users to register and authenticate their devices and configure notification settings. It will provide an event log, which allows users to manage events related to sensors. Additionally, it will have an audit trail of all activity in the system, including user activity. The systems above will be powered by Amazon AWS, it allows us to expand and process

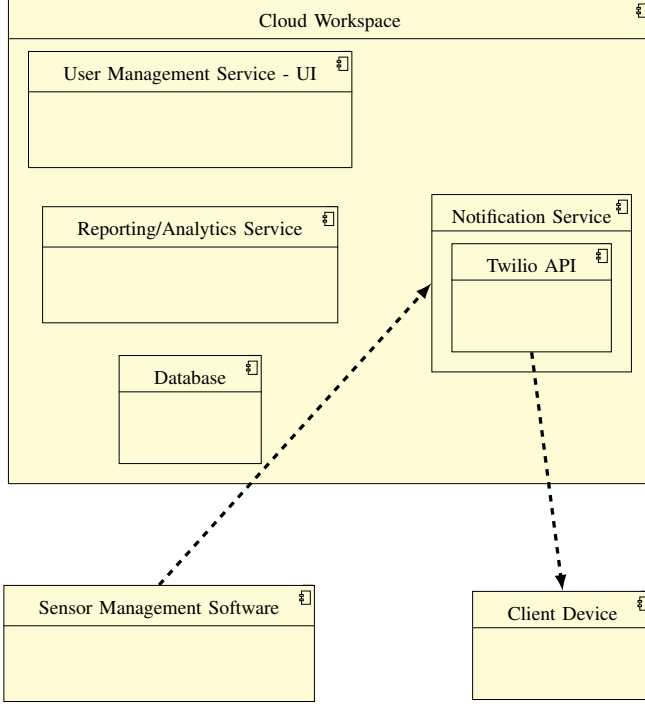


Fig. 3. Software Architecture

the volume of traffic.

The Class Diagram represents the relationships between various classes in the Microwave Motion Security System with SMS Notifications. The User class is associated with the MicrowaveSensor class, which is responsible for detecting motion. The SMSService class is associated with the Database class, which is used to store data related to the SMS services used to send SMS messages to the user. The MicrowaveSensor class is associated with the Notification and SMSService classes, which are responsible for sending notifications and SMS messages to the user when motion is detected. These associations enable the MicrowaveSensor to trigger the Notification and SMSService classes to send a notification and an SMS message to the user, respectively.

The Database class is associated with the Notification, NotificationLog, SMSService, SMSServiceLog, and SensorLog classes, which are used to store data related to notifications, SMS services, and sensor logs. The Notification and SMSService classes have a one-way association with the Database class, indicating that they can use the Database to store data related to the notifications and SMS services sent to the user. Additionally, the Notification and SMSService classes have composition relationships with the NotificationLog and SMSServiceLog classes, respectively. These composition relationships indicate that a Notification or a SMSService “has-a” relationship with their respective log tables, and the

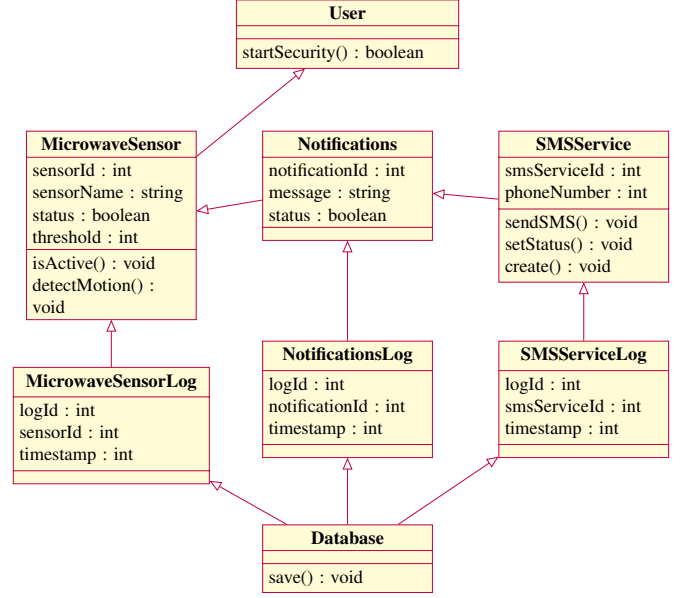


Fig. 4. Software Class Diagram

log tables cannot exist independently of the Notification or SMSService.

Finally, the SensorLog class has a one-way association with the MicrowaveSensor class, which indicates that it can store data related to the logs of the microwave sensors used for detecting motion. Overall, the Class Diagram demonstrates the flow of information and data between the various classes and tables in the system, providing a comprehensive representation of the Microwave Motion Security System with SMS Notifications.

The sequence of events is as follows:

- 1) The User triggers the Microwave Sensor to start monitoring for motion detection by calling the startSecurity() method.
- 2) The MicrowaveSensor detects motion and triggers the Notification and SMSService to send a notification and an SMS message to the user, respectively.
- 3) The Notification creates a new notification by calling the create() method, which generates a new notification ID.
- 4) The Notification saves the notification data to the Database by calling the save(notification) method.
- 5) The Notification sets the status of the notification to “sent” by calling the setStatus“sent” method.
- 6) The SMSService creates a new SMS service by calling the create() method, which generates a new smsServiceId.
- 7) The SMSService saves the SMS service data to the Database by calling the save(sms_service) method.

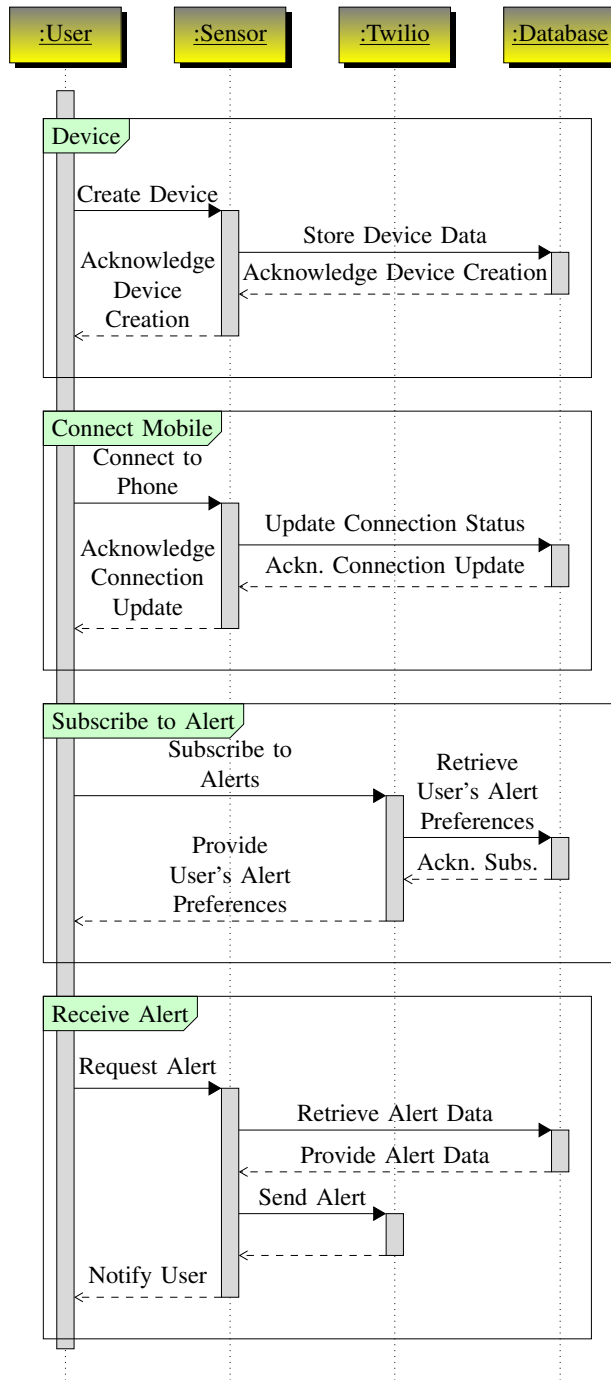


Fig. 5. Software Sequence Diagram

- 8) The SMSService sets the status of the SMS service to “sent” by calling the setStatus“sent” method.
- 9) The SMSService sends an SMS message to the user by calling the sendSMS() method.
- 10) The SMSService saves the log data related to the SMS message to the SMSServiceLog table by calling the save(sms_service_log) method.
- 11) The Notification saves the log data related to the notification to the NotificationLog table by calling the save(notification_log) method.

In this way, the system can detect motion using the MicrowaveSensor, send notifications and SMS messages to the user using the Notification and SMSService classes, respectively, and store data related to these events using the Database, NotificationLog, and SMSServiceLog tables.

B. Design Constraints, Problems, Trade-offs, and Solutions

1) *Design Constraints and Challenges:* Designing a DIY home security system presents several constraints and challenges that require careful consideration. The primary requirement for the system is that it should be easy to use, install, and work seamlessly across a diverse range of devices, which necessitates careful consideration of the user interface and communication protocols. Additionally, the system must be robust and secure to ensure that it is not easily hacked or compromised. This requires implementing encryption and secure data storage methods. Furthermore, the system should have low power consumption, which necessitates the use of sleep modes and energy-efficient components.

Lastly, the design must account for trade-offs, such as balancing the range and sensitivity of the motion sensor with the overall system cost and size. For example, the dimensions of the hardware enclosure should facilitate mounting the module on the doorstep or on the wall at an angle that can provide a coverage angle between a typical visual coverage range of 60 to 75 degrees. However, the dimensions of the hardware enclosure should not be too tall or too wide, as it may cause unequal weight distribution after mounting the enclosure.

2) *Design Solutions and Trade-offs:* An alternative design being explored is to use a different power source, such as a higher-voltage battery or a lower-voltage C/D battery, compared to the current 9V design. However, using a lower voltage battery like AA/AAA would result in less effective mAh and shorter battery life for the system. When using batteries like C/D, another challenge is the additional space it would require in the enclosure, making it heavier and more difficult to mount on walls or ceilings. For this project, 9V batteries offer the best compromise, as they have a reasonable form factor in the shape of a rectangular enclosure with rounded edges, making it easy to design a 3D-printed enclosure with the help of CAD software.

To address the constraints and challenges of designing a DIY home security system, several solutions and trade-offs have been incorporated into the system. The user interface design aims to make it intuitive and user-friendly for users with different technical proficiency levels. Compatibility is achieved by using widely adopted communication protocols and a microcontroller with an integrated Wi-Fi module. To ensure robust security, the system employs contemporary encryption techniques and secure data storage methods, including a secure database hosted on Amazon AWS. In terms of energy efficiency, the device enters sleep mode when not actively detecting motion, reducing power consumption. The system also includes trade-offs, such as optimizing the motion sensor's range and sensitivity, which may impact the overall cost and size of the system. Ultimately, these design solutions and trade-offs contribute to a balanced, effective DIY home security system.

V. SYSTEM IMPLEMENTATION

A. Implementation Overview

In the implementation phase, the group employed a range of technologies including MicroPython, GitHub, Vercel, React, Flask, AWS, and TailwindCSS. This phase encompassed both macOS Sonoma and Windows 10 environments, and it involved various platforms, programming languages, and dependencies.

The group gained practical experience in hardware-related tasks, including wiring and soldering.

B. Implementation of Developed Solutions

Version Control: Git was used for version control, and GitHub facilitated collaboration by allowing team members to join as collaborators. Feature branches were employed for different project segments, and GitHub Issues and Projects were utilized to track tasks, bugs, and project progress.

Code Editors (IntelliJ/VS Code): The team created and configured the project within IntelliJ IDEA, taking advantage of its coding and debugging capabilities. Git and GitHub integration was seamless, enhancing code quality and maintainability with built-in refactoring tools.

Frontend: The frontend was developed using reusable React components for scalability. Data and component communication were managed through React's state and props. Client-side routing was implemented using React Router for a single-page application (SPA) experience. Data from the backend was integrated using asynchronous API calls with tools like fetch or Axios, and component styling was achieved using CSS-in-JS libraries such as Tailwind CSS.

Backend: The team developed APIs with Flask to serve data to the React frontend through defined routes and endpoints. Integration with the MySQL database system was

established. Secure user access was ensured through authentication and authorization using Flask-Login or Flask-JWT. Middleware was utilized for tasks such as logging, error handling, and request/response modification. Reliability was guaranteed by writing unit and integration tests for the Flask application.

Deployment: The React frontend was deployed on web hosting platforms like Vercel, while the Flask backend was hosted on cloud services like AWS. This involved setting up the database, configuring environment variables, and implementing security measures. Testing and deployment were automated using CI/CD pipelines like GitHub Actions for a streamlined development workflow.

C. Implementation Problems, Challenges, and Lessons Learned

Integrating Complexity: Integrating the frontend (React) with the backend (Flask) presented challenges, especially due to differences in technologies, data formats, and API endpoints. Ensuring seamless communication and data flow between the two components required thorough API documentation, cross-functional collaboration, robust error handling, and end-to-end testing to validate data exchange.

Security: Implementing authentication and authorization correctly was crucial but complex. Ensuring the security of user data against potential vulnerabilities like SQL injection and cross-site scripting (XSS) required ongoing education in security best practices, stringent input validation and sanitization, the use of well-established authentication libraries, regular security audits and code reviews, and the implementation of security headers and policies to mitigate potential risks.

Database Management: Setting up and managing a database system presented challenges, particularly with large datasets. Challenges included database migrations, schema design, and optimizing database queries. Addressing these challenges involved dedicating time to robust schema design, implementing automated database backups, using migration tools for schema changes, continuous monitoring and optimization of database queries, and establishing data cleanup routines to maintain database health.

VI. TOOLS AND STANDARDS

A. Tools Used

Throughout the project, a variety of tools were used in the development process to ensure efficient collaboration among team members. The following tools were important in achieving the project goals:

Software

- **IntelliJ IDEA and Visual Studio Code:** Primary IDEs used for coding, debugging, and managing the project

- **Thonny IDE:** Secondary IDE used for coding/debugging hardware programming
- **PyCharm IDE:** Secondary IDE
- **MicroPython:** Employed for programming microcontrollers and enabling embedded system functionalities
- **MySQL:** Relational database management system used to keep track of notification and users
- **ReactJS:** Utilized for building the frontend of the application, providing a dynamic and responsive user interface
- **TailwindCSS:** Utilized for efficient and scalable styling of React components, enhancing the user interface design
- **Python Flask:** Chosen as the backend framework to handle data processing and server-side logic
- **Vercel and AWS:** Respectively used for deploying the frontend and backend on web hosts and cloud services

Hardware

- **Soldering Kit:** Used for assembling the hardware components in a compact configuration
- **Multimeter:** Applied in testing and troubleshooting electrical circuits and components
- **Oscilloscope:** Used for monitoring and analyzing signals within the circuits
- **SolidWorks:** Designing and printing the hardware enclosure
- **Fritzing:** Designing and simulating the hardware assembly

B. Standards

During the course of the project, relatively few established standards were used to define the workflow of the developer team, but there was an emphasis on referencing documentation as much as possible while creating the documentation of the product. Below is a list of the standards used for different aspects of the project:

Software

- **REST (Representational State Transfer):** A standard for designing robust and scalable APIs
- **SHA-2 (Secure Hash Algorithm):** Algorithm used for securing sensitive data of users
- **JWT (JSON Web Tokens):** Algorithm used for secure transmission of information between parties
- **Agile and Scrum Methodologies:** Approaches intended for efficient project management and iterative development
- **Python Flask:** Framework for creation and gateway management of API requests
- **Twilio API:** Responsible for handling notifications sent to mobile devices from sensor, documentation was referenced to set up wireless communication

Hardware

- **802.11xx Wi-Fi:** Wireless network standard used by the microcontroller
- **Raspberry Pi Pico W:** Documentation was heavily used for pinouts and circuit design [6]
- **5.8GHz Microwave Motion Sensor:** Documentation was lightly referenced for pinout in relation to the microcontroller [7]

VII. TESTING AND EXPERIMENTATION

A. Testing and Experiment Scope

With respect to the hardware, there were different approaches to testing and experimentation at different stages of development. The following provides an overview of the different objectives that were present during each stage of development:

- **Ideation:**
 - Maximize performance to power ratio
 - Create a short list of configurations to further test during the assembly process
- **Assembly:**
 - Make use of the breadboard to physically explore configurations
 - Settle on one configuration after testing has been accomplished for the shortlisted configurations
- **Prototyping:**
 - Solder parts and check continuity after soldering
 - Explore soldering parts together to make servicing easier by reducing the number of connections between components.
 - Make light considerations of accessibility/repairability when soldering parts together
- **Finalizing:**
 - Solder parts in a manner that conserves space for the hardware enclosure
 - Make compromise between spatial compression and ease of repair

As far as the software, a systematic approach was taken to ensure the robustness and efficiency of the application. The following objectives guided the testing and experimentation at various stages of development:

- **Research:**
 - Conduct Market Research
 - Identify key challenges to state-of-the-art
 - Conduct contemporary literature review
- **Ideation:**
 - Brainstorm a variety of potential solutions
 - Shortlist a few of the aforementioned solutions
 - Create possible architectures for front and back ends of application
 - Create low-fidelity proof-of-concept

- **Implementation:**

- Choose one configuration of tools and technologies
- Create high-fidelity prototype by implementing certain features in an agile fashion
- Leverage developers to test for function first, then optimization second

- **Benchmarking:**

- Emphasize optimization of the code and robustness
- Achieve a code coverage of at least 80 percent
- Reduce test case escapes to zero

- **Finalizing:**

- Make cosmetic changes to the application
- Create script for live demonstration rehearsal
- Repeatedly test the system to ensure stability and consistency over time.

B. Testing and Experiment Approach

Testing the hardware was an iterative process that entailed using simulations, continuity checks, and constant soldering of different parts in different ways to conserve space to make the hardware as physically unobtrusive as possible while keeping it easy to repair. During the ideation phase, simulations were used with the Fritzing software where different power deliveries and wiring configurations were tested via virtual simulations. This led to some of the compromises made as described in **Design Constraints, Problems, Trade-offs, and Solutions**.

During the assembly phase, a breadboard was used to test without spending time and effort soldering and desoldering the configuration. Once the hardware configuration reached a satisfactory level of performance, only then were the parts soldered together. During the prototyping phase, different wires and connectors were used to determine the best manner of mounting different components to one continuous connection. A proof of concept was made that had no consideration for conservation of space, and continuity checks were done with a multimeter to ensure that each solder connection was made properly.

In finalizing the hardware configuration, several spare parts were soldered together and connected onto the board that used different techniques to conserve space. Again, the solder joints were tested for continuity with a multimeter. In addition to conserving space, an additional consideration was made to avoid short-circuits by exploring both electrical tape and heatshrink. From there, ease of repair was taken into account by documenting the nature of the solder joints so end users could easily replicate the consolidation done by the hardware system integrators.

In testing the software implementation, the primary goal was to validate the functionality, reliability, and performance of the system across various components and integrations.

The testing process was comprehensive, covering both unit and integration levels to ensure the smooth functioning of individual elements as well as their seamless integration into the whole system.

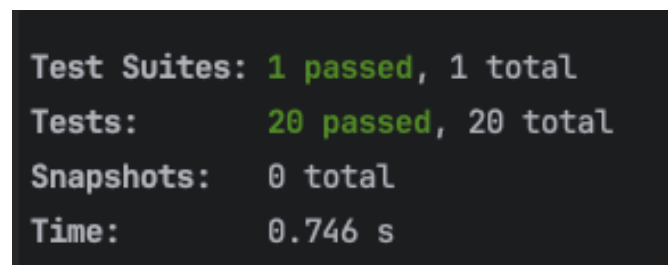
The unit testing phase involved meticulously testing each system component in isolation, scrutinizing the functions and methods within the Python and Flask backend, along with the ReactJS frontend. The developers conducted a thorough examination of the functionality of MySQL database interactions and confirmed the dependable data transmission between the microcontroller and the backend server using MicroPython.

Integration testing was then conducted to secure a seamless connection between the frontend, backend, and database, which guaranteed a consistent and accurate flow of data. Validation was done by examining the communication efficiency between the microcontroller, sensors, and backend server, ensuring that the entire system operated seamlessly when all components were brought together.

C. Testing and Experiment Results and Analysis

While the majority of test cases passed successfully, demonstrating the system's robustness and reliability, any failed tests were promptly addressed, with bugs fixed and tests rerun until success was achieved.

- **Performance Test Result Analysis:** The system performed well under normal conditions and was able to handle load up to performance benchmarks.
- **Test Coverage:** To ensure confidence in the system's reliability, tests achieved high coverage across the code-base.
- **Bug Distribution Report:** Bugs were categorized based on their severity, with critical and major bugs given priority in fixing. GitHub Issues detailing of bugs, their severity, and the components they affected was created.



```

Test Suites: 1 passed, 1 total
Tests:      20 passed, 20 total
Snapshots:  0 total
Time:       0.746 s
  
```

Fig. 6. Using JMeter Test Suite

By rigorously following this comprehensive testing and experiment approach, the developers ensured that the system was reliable, performed well, and met all the defined requirements and benchmarks.

| Average | Min | Max | Std. Dev. | ... | Throug... ↑ | Received ... |
|---------|-----|-----|-----------|-----|-------------|--------------|
| 5 | 0 | 26 | 3.62 | ... | 5.0/sec | 1.99 |
| 5 | 0 | 26 | 3.62 | ... | 5.0/sec | 1.99 |

Fig. 7. JMeter 100 Request Benchmark Metrics

| Filters | Q is:issue is:open |
|---|--------------------|
| 8 Open | 3 Closed |
| Update GitHub to include MicroPython | enhancement |
| #17 opened 40 minutes ago by shohinsan | |
| Improve Backend Req/Res for better Performance | bug |
| #16 opened 42 minutes ago by shohinsan | |
| Include Alert Tables in Frontend | enhancement |
| #15 opened 43 minutes ago by shohinsan | |
| Finalize Backend and Dockerize (Easier To Demonstrate @ Expo) Without | discussion |
| #11 opened on Jun 23 by shohinsan | |
| Design Hardware To Work With Backend | enhancement |
| #10 opened on Jun 23 by shohinsan | |
| Update README | documentation |
| #9 opened on Jun 23 by shohinsan | |
| Document Gantt Chart to LaTeX | documentation |
| #8 opened on Jun 23 by shohinsan | |
| Implement CI/CD workflows After Auth Done | enhancement |
| #5 opened on Jun 15 by shohinsan | |

Fig. 8. Issue Tracking System

VIII. CONCLUSION AND FUTURE WORK

The MorteSense DIY security system represents a significant step forward in providing accessible and reliable security solutions for both residential and commercial properties. By integrating microwave motion sensors and SMS notifications, the system offers an affordable and user-friendly approach to monitoring and securing properties with ease of repair and maintenance in mind.

The project ensured the creation of an intuitive user interface, compatibility with a variety of devices, and the establishment of a robust and secure system that is easy to install and operate. The system's focus on real-time alerts through SMS notifications enables prompt responses to potential security breaches, ultimately contributing to enhanced safety and security for homeowners and businesses alike.

The project's approach to leveraging contemporary micro-controllers and Wi-Fi connectivity, along with its modular

design principles, underscores the system's adaptability and potential for future expansion. Furthermore, the integration of a user-friendly web interface and the incorporation of Twilio API for SMS notifications contribute to the overall accessibility and sustainability of the system. By considering factors such as user behavior, power outages, and software compatibility, the system was designed to mitigate these risks and ensure a robust security solution for end-users.

The comprehensive documentation, including the PCB diagram and Class Diagram, highlights the meticulous planning and execution involved in the hardware and software integration. This careful attention to detail ensures a seamless data flow and an effective user experience, further emphasizing the system's reliability and user-friendliness.

Additionally, the incorporation of advanced technologies such as blockchain and machine learning, while beyond the current scope, presents promising avenues for further research and development in the field of smart home security systems.

In summary, the MorteSense DIY security system's innovative use of technology, combined with its emphasis on accessibility and user-friendliness, makes it a valuable and practical solution for those seeking openly-documented security measures for their homes and businesses.

ACKNOWLEDGMENTS

The authors would like to express their gratitude to the San Jose State University (SJSU) Library Student Computing Services, the SJSU Library Makerspace, and the Society of Computer Engineers chapter at SJSU.

In addition to the institutions listed above, the authors would like to thank Dr. Faramarz Mortezaie for his role as the project advisor.

REFERENCES

- [1] Q. I. Sarhan, "Systematic Survey on Smart Home Safety and Security Systems Using the Arduino Platform," *IEEE Access*, vol. 8, pp. 128 362–128 384, 2020.
- [2] S. Arif, M. A. Khan, S. U. Rehman, M. A. Kabir, and M. Imran, "Investigating Smart Home Security: Is Blockchain the Answer?" *IEEE Access*, vol. 8, pp. 117 802–117 816, 2020.
- [3] M. A. Khan, S. Abbas, A. Rehman, Y. Saeed, A. Zeb, M. I. Uddin, N. Nasser, and A. Ali, "A Machine Learning Approach for Blockchain-Based Smart Home Networks Security," *IEEE Network*, vol. 35, no. 3, pp. 223–229, Jun. 2021.
- [4] N. Komninos, E. Philippou, and A. Pitsillides, "Survey in Smart Grid and Smart Home Security: Issues, Challenges and Countermeasures," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1933–1954, 2014.
- [5] A. C. Jose and R. Malekian, "Improving Smart Home Security: Integrating Logical Sensing Into Smart Home," *IEEE Sensors Journal*, vol. 17, no. 13, pp. 4269–4286, Jul. 2017.
- [6] "Raspberry Pi Pico W Documentation." [Online]. Available: <https://www.raspberrypi.com/documentation/microcontrollers/raspberry-pi-pico.html>

- [7] "CQRobot - 5.8 GHz Doppler Effect Microwave Motion Sensor Documentation," Dec. 2022. [Online]. Available: http://www.cqrobot.wiki/index.php/5.8GHz_Doppler_Effect_Microwave_Motion_Sensor_SKU:_CQRSENB02

BIOGRAPHY

Shohin Abdulkhamidov - Shohin Abdulkhamidov is an aspiring software engineer with a focus on Fullstack development. Currently pursuing a degree in software engineering at San Jose State University, who is set to graduate with a strong foundation in both front-end and back-end technologies. Eager to contribute to the tech industry, Shohin is actively seeking job opportunities that allow him to apply his expertise in Fullstack development and make a meaningful impact.

Diego Cruz - Diego R Cruz is a software engineering undergraduate at San Jose State University, who is actively pursuing opportunities in Human-Computer Interaction. Throughout this project, Diego served as a project manager, hardware engineer, and was responsible for documentation surrounding the project. Actively exploring different technologies such as audio post-processing and realtime audio processing, Diego remains dedicated to learning about user interface design, user interaction design, and front-end development.

Diego Garcia-Carrasco - Diego Garcia-Carrasco, a software engineering undergraduate at San Jose State University, has significantly advanced his expertise through internships and mentorships at notable tech companies like CDW, Google, and LinkedIn. These experiences have not only involved him in groundbreaking AI leadership programs but also deepened his interest in Cloud Computing and Distributed Systems. Actively contributing to the advancement of Artificial Intelligence, Diego's portfolio, which encompasses both iOS and web development, reflects his dedication to driving technological innovation and fostering community engagement in the tech sector.

Spartak Gevorgyan - Spartak Gevorgyan, an undergraduate student in software engineering at San Jose State University, has significantly honed his skills through internships and mentorships with renowned tech firms, including Ansys. These experiences have not only immersed him in pioneering engineering simulation initiatives but have also deepened his fascination with CAE and Multiphysics Engineering Simulation technologies. Spartak's diverse portfolio, covering both web and software development, is a testament to his dedication to propelling technological advancements and fostering community engagement within the tech sector.

Dr. Faramarz Mortezaie - The one and only MorteSensi

APPENDIX A OVERVIEW OF SMART HOME ARCHITECTURE [4]

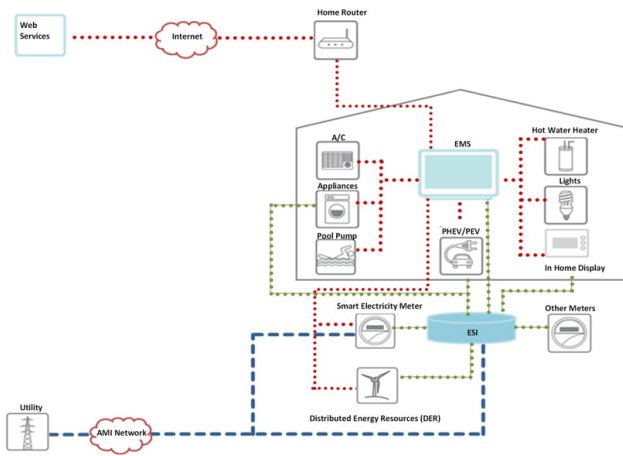


Fig. 9. An overview of a smart home's architecture, internal and external environments

APPENDIX B
FLOWCHART OF DOOR STATE CHANGES AND SENSOR
OPERATIONS [5]

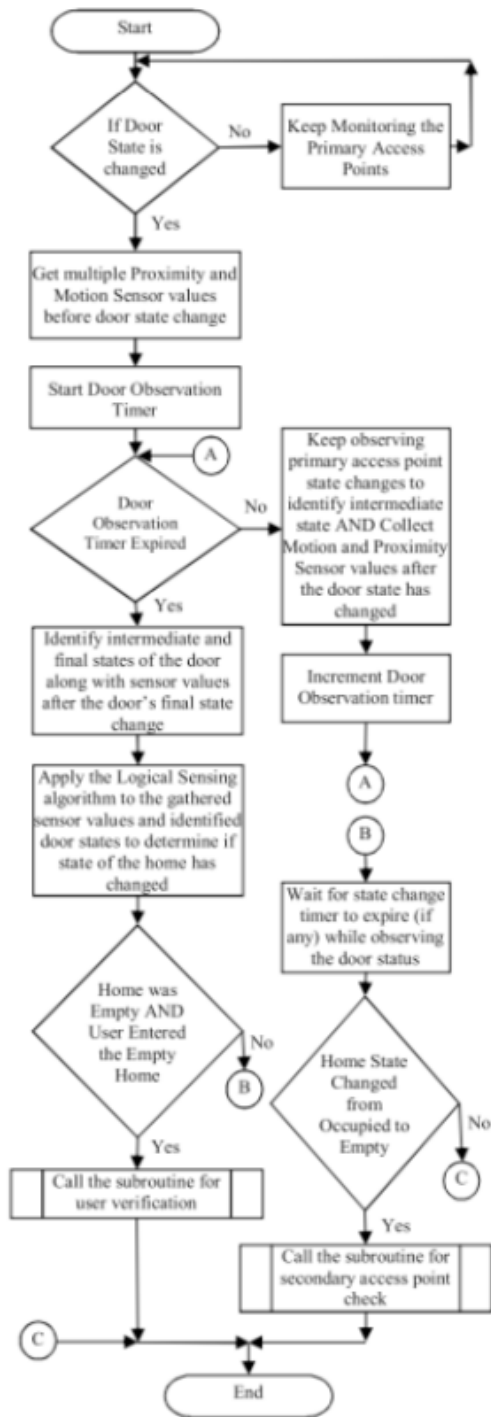


Fig. 10. Flowchart showing door state changes and sensor operations of the primary access point

APPENDIX C
FLOORPLAN AND SENSOR DEPLOYMENTS [5]

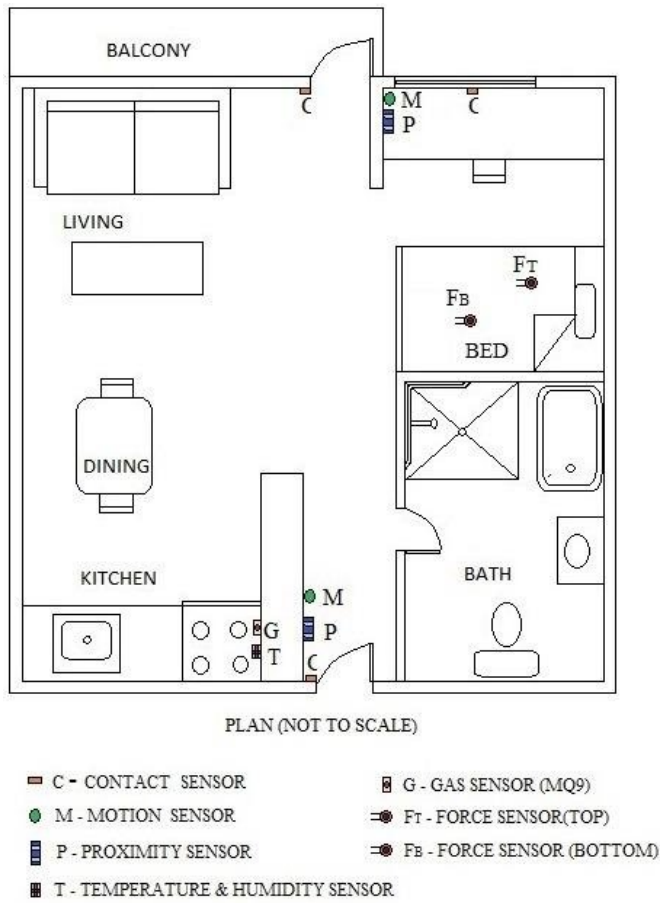


Fig. 11. Plan and location of the sensor deployments in the apartment

APPENDIX D PROCESS FLOW OF SMART HOME ARCHITECTURE WITH BLOCKCHAIN [2]

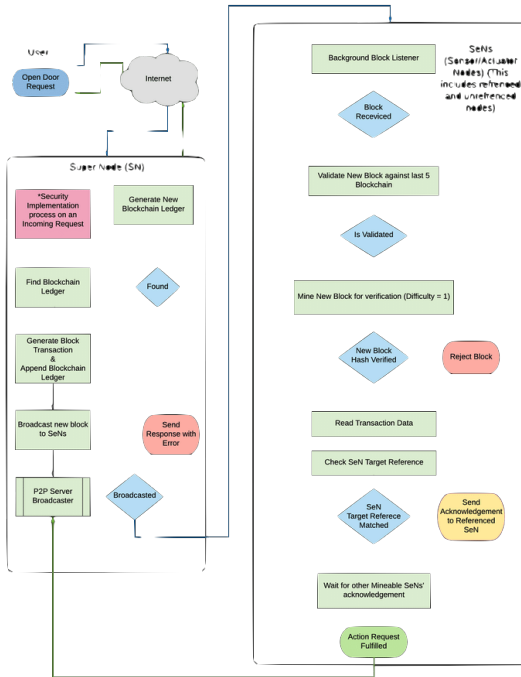


Fig. 12. Process flow of the proposed architecture

APPENDIX E
FRAMEWORK OF SMART HOME WITH DELM
TECHNOLOGY [3]

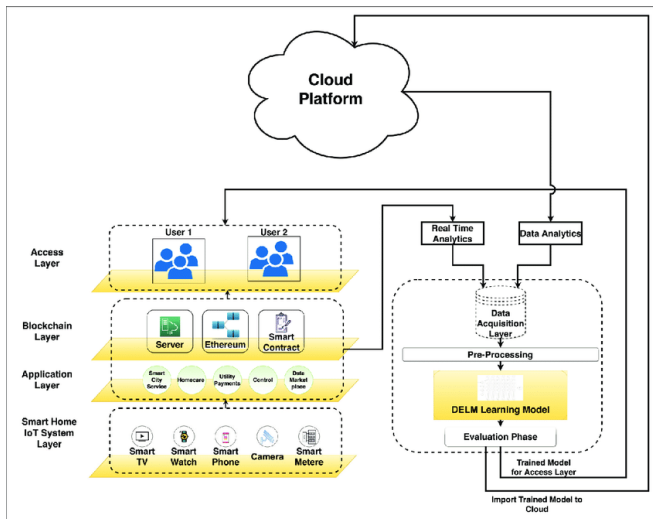


Fig. 13. A 4-layer application framework of a blockchain-based smart home empowered with DELM